Documento de arquitectura

MISW 4204 – Desarrollo de software en la nube
Simón Buriticá
Jhonn Sebastián Calderón Bravo
Diego Andrés Naranjo Ríos
Juan Pablo Rodríguez García

Universidad de los Andes – 2024

Diagrama de clases:

En la figura 1 se muestra el diagrama de clases obtenido a través de la lectura expuesta por la IDRL para la competencia. Se exponen 3 clases principales las cuáles son “usuario”, que representa al piloto que quiere participar en la competición, “video” que interacciona directamente con el usuario ya que un mismo piloto puede tener ningún o muchos videos, y “task” que representan las tareas de edición de los videos. Adicionalmente, se tiene una enumeración que identifica el estado de las tareas.

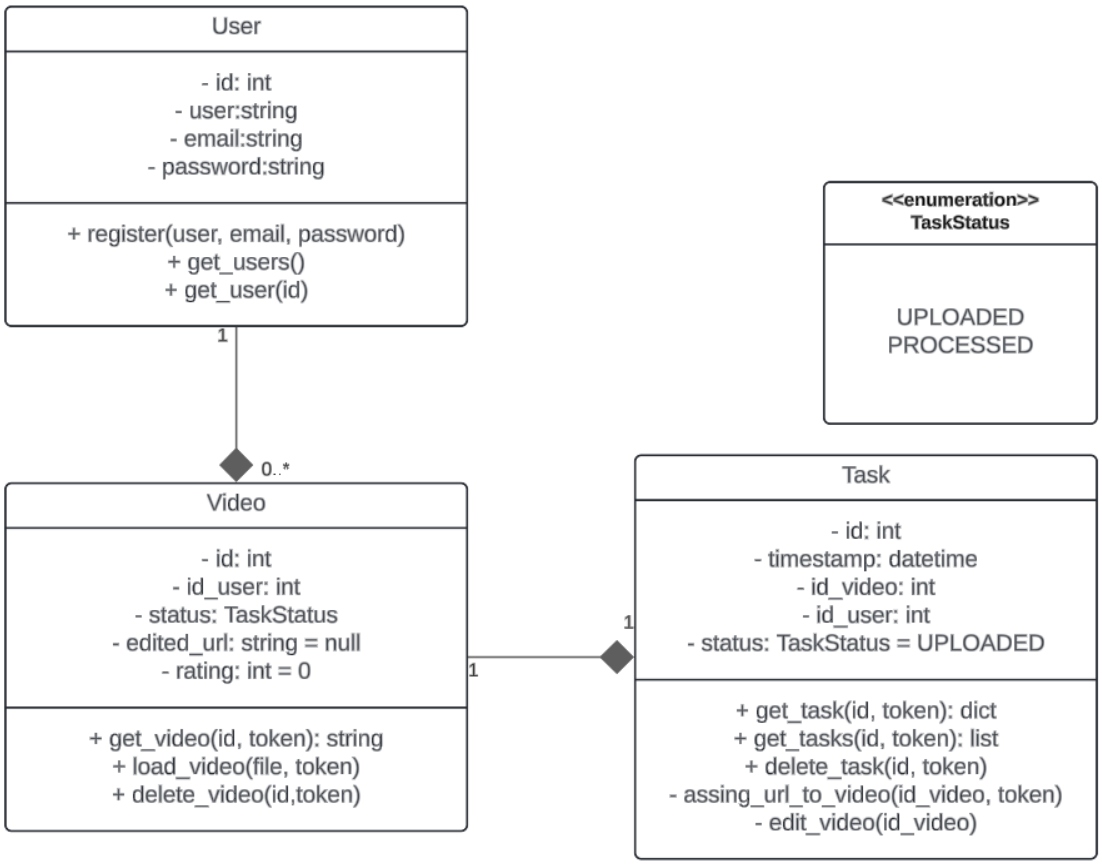

Figura 1 - Diagrama de clases preliminar a la construcción

Diagrama funcional:

Para la solución del proyecto se construyó un diagrama de componentes que tiene las siguientes características:

- Un proxy inverso que recibe los estímulos de los usuarios y se encarga de comunicarlos a la API REST. Este también cumple la función de gestionar múltiples peticiones y actuar como un balanceador de carga, al igual que enmascarar los métodos y atributos del backend, dando una capa extra de seguridad al desarrollo
- Una API Gateway que desacopla el llamado a los endpoints de cada modelo. Proporciona una interfaz unificada para el consumo de servicios. También es responsable de realizar las solicitudes a los servicios internos del sistema. Al ejercer esta función centraliza la lógica relacionada con la gestión de solicitudes.
- El autorizador que es responsable de gestionar la autenticación y la autorización de los usuarios dentro del sistema. Genera y valida tokens de acceso para los usuarios autenticados, garantizando que solo los que tengan permiso adecuado puedan acceder a los recursos.
- Una clase que se encarga de gestionar la información de los usuarios registrados en la plataforma. Esta clase facilita la interacción entre los usuarios y el sistema.
- Las clases principales del modelo lógico, estas clases representan las entidades fundamentales dentro del modelo de datos de la aplicación. Son responsables de definir la estructura y el comportamiento de los objetos principales con los que trabaja el sistema, lo que incluye la lógica de negocio y las relaciones entre las distintas entidades
- Un message broker que recibe los eventos de creación de tareas y actúa como un canal de comunicación asíncrona para llevar estos eventos al video worker. En este caso se utilizó a Google Pub/Sub para esta función.
- Un componente llamado “video worker”, el cual se encarga de desencolar los mensajes, inicializar las tareas de edición de los videos, y enviar la petición al componente video para enlazar el video editado
- El File Server es un componente dedicado al almacenamiento y gestión de archivos multimedia, incluyendo tanto los videos originales como los editados dentro del sistema. Para esto se utiliza un bucket de Google Storage.

En la figura 2, es posible ver el diseño propuesto, el cual está enfocado en ser un sistema desacoplado con alta modificabilidad, disponible y seguro

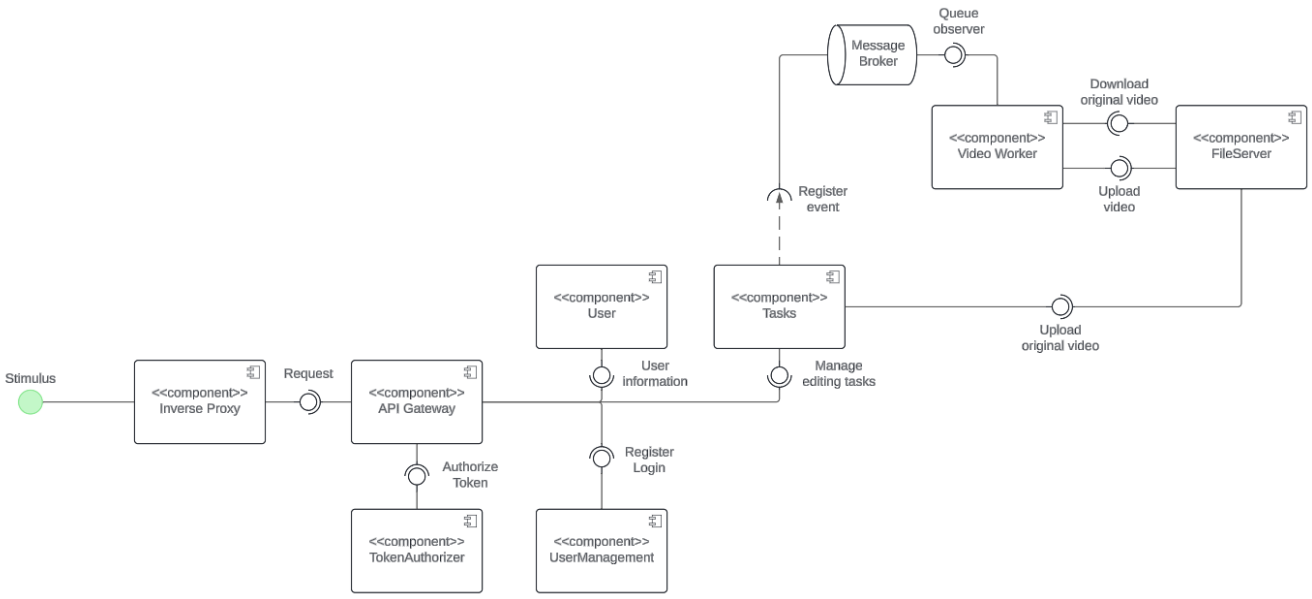

Figura 2 - Diagrama funcional para la solución del proyecto

Diagrama de despliegue

El despliegue de la solución se muestra en la figura 3. Para la versión actual del sistema se realizó un diagrama de despliegue UML con todos los servicios de Google Cloud Platform (GCP) que se deben utilizar para la solución propuesta. El listado de servicios utilizados de GCP son los siguientes:

- **Cloud Load Balancing:** Este servicio es el encargado de recibir todas las peticiones externas desde el cliente y es la puerta de entrada a la aplicación. Se comunica directamente con los diferentes proxy de las instancias en la capa web y distribuye la carga entre ellas

- **Compute Engine:** Utilizado para las diferentes máquinas virtuales que requieren las diferentes capas de la aplicación. En el caso de la arquitectura propuesta se cuentan con mínimo 2 máquinas virtuales, 1 para la capa web y otra para la capa worker. Estas están configuradas para escalar de forma horizontal por medio de la configuración de instance groups. Para el caso de la capa web, esta escalará de uno en uno hasta un máximo de 3 instancias distribuidas en múltiples zonas de la región us-central-1 cuándo se supere un uso de CPU en el grupo de instancias de 40%. Para el caso de la capa worker, esta escalará de uno en uno cuándo el número de mensajes en la cola esperando ser procesados supere los 15 por instancia de compute engine

- **Cloud Storage:** Este servicio almacena por medio de buckets los videos que carga el piloto para ingresar a la competición. También almacena los videos editados por la capa worker

- **PubSub:** El servicio de PubSub reemplaza el message broker que se tenía anteriormente por medio del software RabbitMQ por un servicio en la nube escalable y que permite la configuración de múltiples worker de forma sencilla. Su función es recibir las ordenes de trabajo creadas por la capa web cuándo un usuario carga un video y redistribuirlas a las diferentes instancias de la capa worker para iniciar los procesos de edición de video

- **Cloud SQL:** Es el servicio de base de datos relacionales de GCP. Se utiliza para guardar la información de los usuarios, la identificación y ruta de los videos, la trazabilidad de las ordenes de trabajo de las tareas encoladas y los logs de las instancias de la capa worker, que incluyen errores y tiempos de procesamiento

- **Virtual Private Cloud:** Todos los servicios contenidos se encuentran aislados por medio de una VPC con un conjunto de subredes ubicadas en la región us-central-1. Esto, en conjunto con reglas en **Cloud Firewall Rules** permiten garantizar la seguridad del sistema, al igual que crear un flujo de trabajo ordenado entre los diferentes servicios de la aplicación

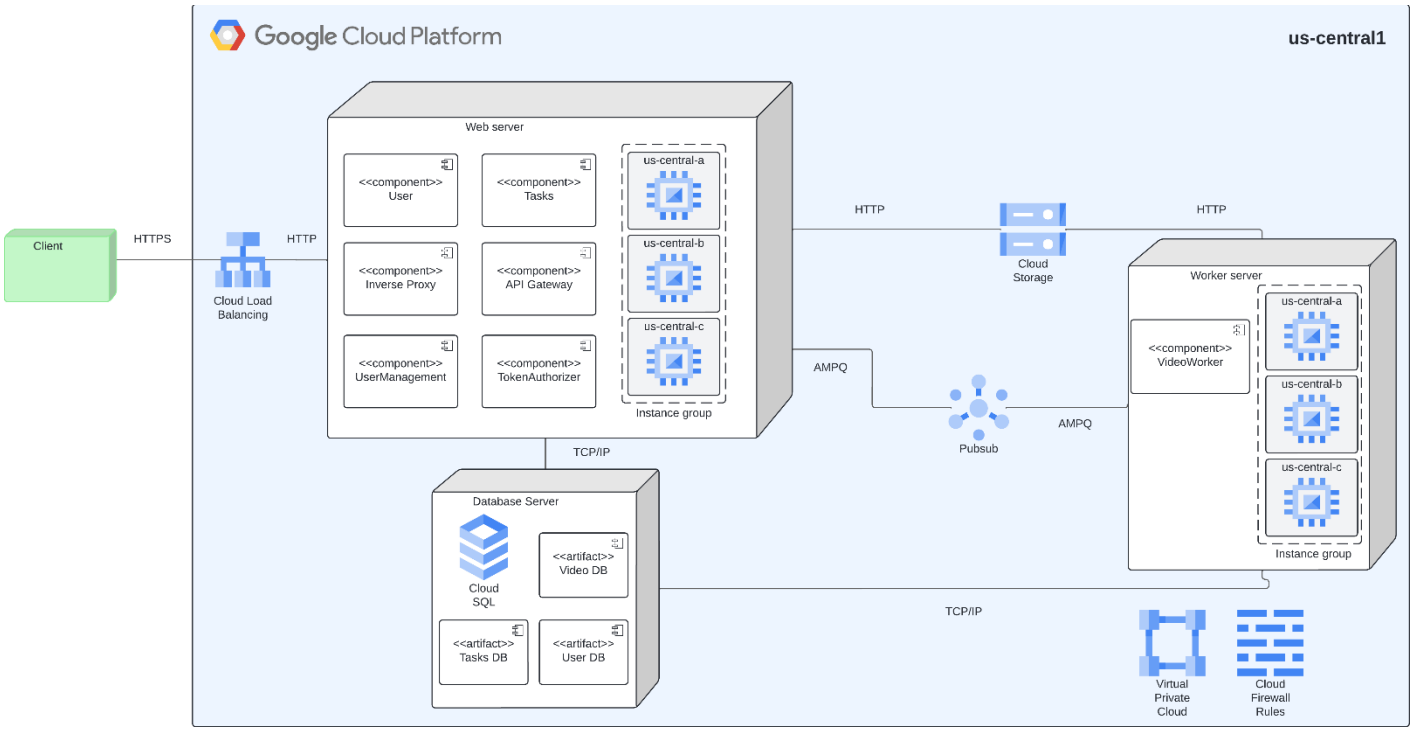

Figura 3 - Diagrama de despliegue propuesto. En él se muestran los diferentes nodos

El diagrama muestra la distribución de los servicios de acuerdo con cada nodo. En esta arquitectura, los nodos de la capa web y worker son replicables por medio de los grupos de instancia y se distribuyen en las zonas asignadas en el diagrama. Se juntan todas en el mismo nodo a manera de demostración y es necesario aclarar que cada nodo de componentes realmente solo representa una instancia del conjunto de instancias.